Realtime publishers

The Shortcut Guide™ To

Architecting iSCSI Storage for Microsoft Hyper-V

sponsored by

Greg Shields

Copyright Statement

Chapter 2: Creating Highly-Available Hyper-V with iSCSI Storage

It's worth saying again that Hyper-V alone is exceptionally easy to set up. Getting the basics of a Hyper-V server up and operational is a task that can be completed in a few minutes and with a handful of mouse clicks. But in the same way that building a skyscraper is so much more than welding together a few I-beams, creating a production-worthy Hyper-V infrastructure takes effort and planning to be successful.

The primary reason for this dissonance between "installing Hyper-V" and "making it ready for operations" has to do with high availability. You can think of a Hyper-V virtual infrastructure in many ways like the physical servers that exist in your data center. Those servers have high-availability functions built-in to their hardware: RAID for drive redundancy, multiple power supplies, redundant network connections, and so on. Although each of these is a physical construct on a physical server, they represent the same kinds of things that must be replicated into the virtual environment: redundancy in networking through multiple connections and/or paths, redundancy in storage through multipathing technology, redundancy in processing through Live Migration, and so on.

Using iSCSI as the medium of choice for connecting servers to storage is fundamentally useful because of how it aggregates "storage" beneath an existing "network" framework. Thus, with iSCSI it is possible to use your existing network infrastructure as the transmission medium for storage traffic, all without needing a substantially new or different investment in infrastructure.

To get there, however, requires a few new approaches in how servers connect to that network. Hyper-V servers, particularly those in clustered environments, tend to make use of a far, far greater number of network connections than any other server in your environment. With interfaces needed for everything from production networking to storage networking to cluster heartbeat, keeping straight each connection is a big task.

This chapter will discuss some of the best practices in which to connect those servers properly. It starts with a primer on the use of the iSCSI Initiator that is natively available in Windows Server 2008 R2. You must develop a comfort level with this management tool to be successful, as Hyper-V's high level of redundancy requirements means that you'll likely be using every part of its many wizards and tabs. In this section, you'll learn about the multiple ways in which connections are aggregated for redundancy. With this foundation established, this chapter will continue with a look at how and where connections should be aggregated in single and clustered Hyper-V environments.

The Windows iSCSI Initiator: A Primer

Every iSCSI connection requires two partners. On the server is an iSCSI *initiator*. This initiator connects to one or more iSCSI *targets* that are located on a storage device elsewhere on the network. In order to create this connection, software is required at both ends. At the target is software that handles incoming connections, directs incoming initiator traffic to the right LUN, and ensures that security is upheld between LUNs and the initiators to which they are exposed.

Each computer connecting to that iSCSI storage device needs its own set of initiator software to manage its half of the connection. Native to Windows Server 2008 R2 (as well as previous editions, although we will not discuss them in this guide) is the Windows iSCSI Initiator. Its functionality is accessed through a link of the same name that is found under Administrative Tools. Figure 2.1 shows an example of the iSCSI Initiator Control Panel as seen in Windows Server 2008 R2. Here, three separate LUNs have been created using the iSCSI target device's management toolset and exposed to the server.

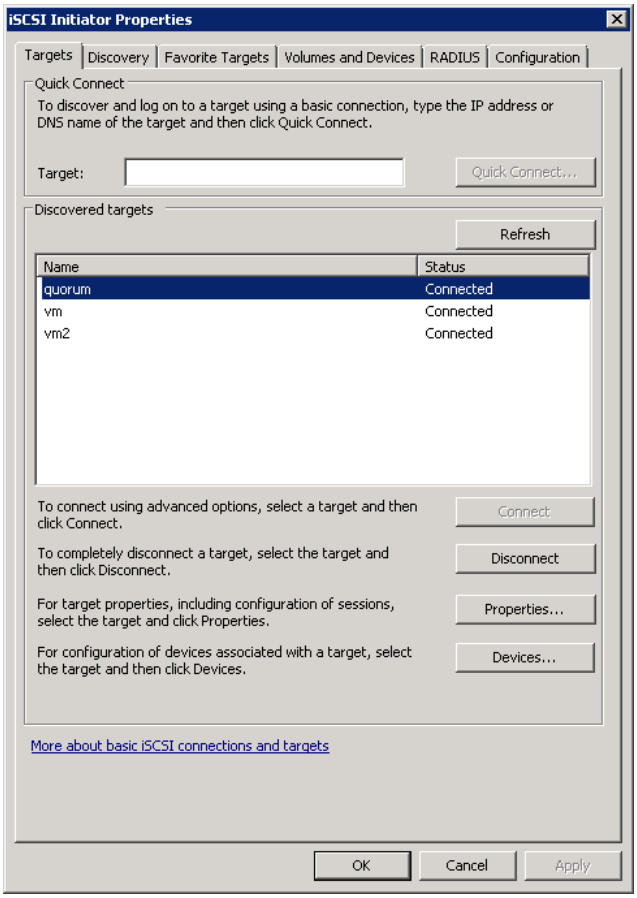

Figure 2.1: The default iSCSI Initiator.

Note

Storage connections are generally made first using the storage device's management toolset. They are exposed via a fully-qualified domain name or IP address to one or more servers that may or may not share concurrent access.

The process to accomplish this first step is different based on the storage device used. Each storage device vendor develops their own toolset for accomplishing this task, with some leveraging Web-based utilities while others use client-based utilities. Consult your vendor's administrative guide for details on this process.

In the simplest of configurations, connecting an iSCSI target to an initiator is an exceptionally easy process. First, enter the target DNS name or IP address into the box labeled Target, and click Quick Connect. If settings have been correctly configured at the iSCSI SAN, with LUNs properly exposed to the server, a set of targets should appear in the box below. Selecting each target and clicking Connect will enable the connection.

At this point, the disk associated with that target's LUN will be made available within the server's Disk Management console. There, it will need to be brought online, initialized, and formatted to make it useable by the system.

From an architectural standpoint, a number of on-system components must work in concert for this connection to occur. As Figure 2.2 shows, the operating system (OS) and its applications leverage the use of a vendor-supplied disk driver to access SCSI-based disks. The SCSI layer in turn wraps its commands within iSCSI network traffic through the iSCSI Initiator, which itself resides atop the server's TCP/IP communication. Incoming traffic arrives via a configured NIC, and is recompiled into SCSI commands, which are then used by the system for interaction with its disks.

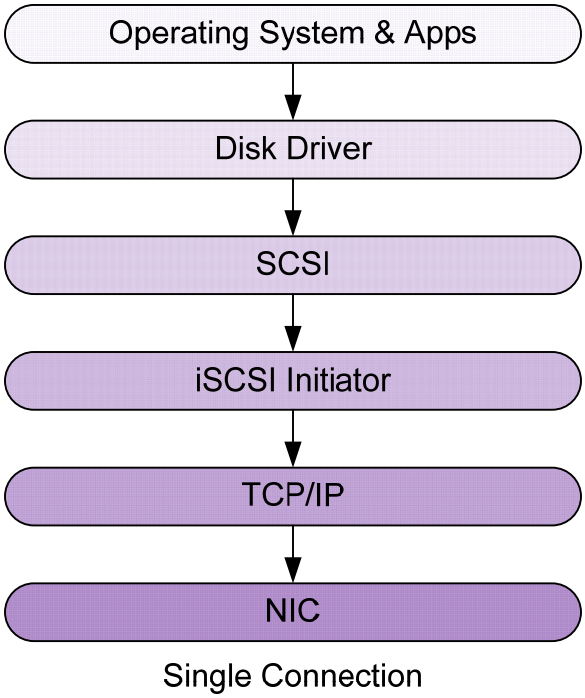

Figure 2.2: Multiple elements work together to enable an OS to interact with iSCSI disks.

This overt simplicity in configuring LUNs with a single path belies the added complexity when adding additional connections. You'll find that once your number of storage connections grows beyond one per server, your number of high-availability options and configurations within each option grows dramatically. To assist, let's take a look through the three kinds of high-availability options that are commonly considered today.

NIC Teaming

One option for high availability that is often a first consideration by many administrators is the use of NIC teaming at the server. This option is often first considered because of the familiarity administrators have with NIC teaming over production networks. Using this method, multiple NICs are bonded together through the use of a proprietary NIC driver. In Figure 2.2, this architecture would be represented by adding additional arrows and NIC cards below the element marked TCP/IP.

Although teaming or bonding together NICs for the purpose of creating storage connection redundancy is indeed an option, be aware that this configuration is *neither supported nor considered a best practice* by either Microsoft or most storage vendors. As such, it is not an option that most environments should consider for production deployments.

Although NIC teaming provides the kind of redundancy that works for traditional network connections, two alternative protocols have been developed that accomplish the same goal but with better results. Multipath Input/Output (MPIO) and Multiple Connections per Session (MPS) are two very different protocols that enable multiple connections between servers and storage and are designed specifically to deal with the needs of network storage traffic.

MCS

The first of these protocols is MCS. This protocol operates at the level of the iSCSI initiator (see Figure 2.3) and is a part of the iSCSI protocol itself, defined within its RFC. Its protocol-specific technology enables multiple, parallel connections between a server and an iSCSI target. As a function of the iSCSI initiator, MCS can be used on any connection once the iSCSI initiator is enabled for use on the system.

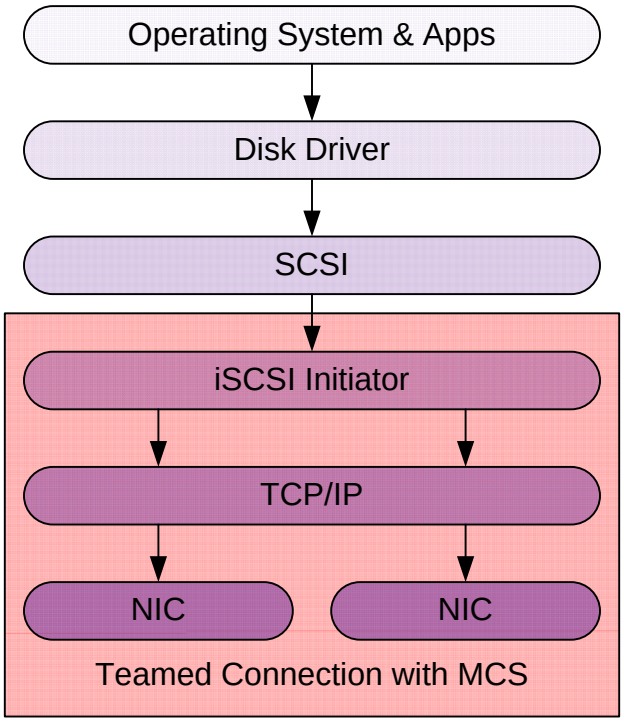

Figure 2.3: Teaming connections with MCS.

To enable MCS for a storage connection, connect first to a target and then click Properties within the Targets tab of the iSCSI Initiator Control Panel. In the resulting screen, click the button marked MCS that is found at the bottom of the wizard. With MCS, multiple initiator IP addresses are connected to associated target portal IP addresses on the storage device (see Figure 2.4). These target portal IP addresses must first be configured on both the server and the storage device prior to connecting an initiator.

Unlike MPIO, which will be discussed next, MCS does not require any special multipathing technology to be coded by the manufacturer and installed to connecting servers; however, support must be available on the storage device. Consult your manufacturer's documentation to verify whether MCS support is available within your storage hardware.

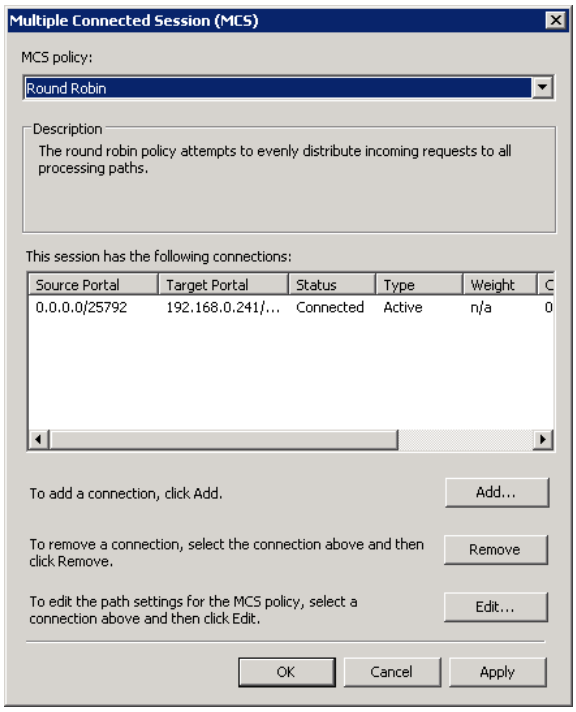

Figure 2.4: Configuring MCS for a storage connection.

MCS can be configured with one of five different policies, with each policy determining the behavior of traffic through the connection. Policies with MCS are configured per session and apply to all LUNs that are exposed to that session. As such, individual sessions between initiator and target are given their own policies. The five policies function as follows:

- *Fail Over Only.* With a failover policy, there is no load balancing of traffic across the session's multiple connections. One path is used for all communication up until the failure of that path. When the active path fails, traffic is then routed through the standby path. When the active path returns, routing of traffic is returned back to the original path.
- *Round Robin.* Using Round Robin, all active paths are used for routing traffic. Using this policy, communication is rotated among available paths using a round robin approach.

- *Round Robin with a subset of paths.* This third policy operates much like the Round Robin policy, with one important difference. Here, one or more paths are set aside as standby paths to be used similar to those in the Fail Over Only policy. These paths remain in standby until a primary path failure occurs. At that point, the standby path is used in the Round Robin with the surviving paths. When the failed primary path returns, traffic is routed again through that path, returning the subset path to standby.
- *Least Queue Depth.* The Least Queue Depth policy functions similarly to Round Robin, with the primary difference being in the determination of how traffic is load balanced across paths. With Least Queue Depth, each request is sent along the path that has the least number of queued requests.
- *Weighted Paths.* Weighted Paths provides a way to manually configure the weight of each path. Using this policy, each path is assigned a relative weight. Traffic will be balanced across each path based on that assigned weight.

MPIO

Another option for connection redundancy is MPIO. This protocol accomplishes the same functional result as MCS but uses a different approach. With MPIO (see Figure 2.5), disk manufacturers must create drivers that are MPIO-enabled. These disk drivers include a Device-Specific Module (DSM) that enables the driver to orchestrate requests across multiple paths. The benefit with MPIO is in its positioning above the SCSI layer. There, a single DSM can be used to support multiple network transport protocols such as Fibre Channel and iSCSI.

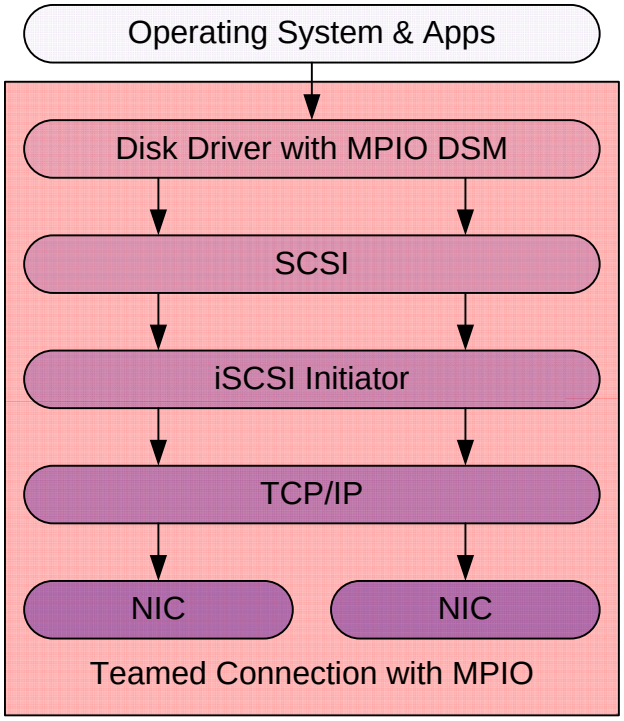

Figure 2.5: Teaming connections with MPIO.

Your hardware manufacturer's DSM must be installed to each server where you intend to use MPIO. Alternatively, a default DSM is available in Windows Server 2008 R2 that functions with many storage devices. Consult your manufacturer's documentation to verify whether their separate vendor driver installation is required, or if the default Windows DSM is supported.

To use the default DSM with iSCSI storage, two steps are necessary. First, install the Multipath I/O Feature from within Server Manager. Installing this feature requires a reboot and makes available the MPIO Control Panel within Administrative Tools. Step two involves claiming all attached iSCSI devices for use with the default Microsoft DSM. Do this by launching the MPIO Control Panel and navigating to the Discover Multi-Paths tab. There, select the *Add support for iSCSI devices* check box and reboot the computer. This process instructs the server to automatically claim all iSCSI devices for the Microsoft DSM, regardless of their vendor or product ID settings.

Once enabled, MPIO is configured through the iSCSI Initiator Control Panel. There, select an existing target and click Connect. In the resulting screen, select the *Enable multi-path* check box and click Advanced. The Advanced settings screen for the connection provides a place where additional initiator IP addresses are connected to target portal IP addresses. Repeating this process for each source and target IP address connection will create the multiple paths used by MPIO.

Verifying path creation is accomplished by selecting an existing target and clicking Devices and then MPIO. The resulting screen, seen in Figure 2.6, displays the configured paths from the server to the target. Also in this location is the selection for configuring the load-balance policy for the LUN.

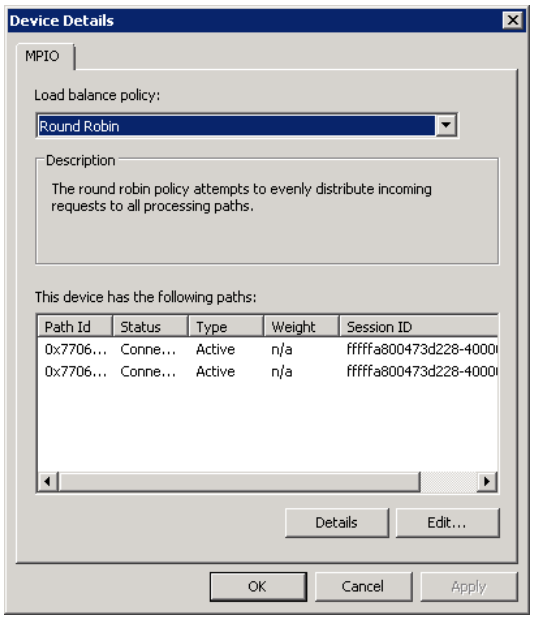

Figure 2.6: Setting an MPIO load-balance policy.

MPIO in Windows Server 2008 R2 can use any of the same five policies as MCS as well as one additional policy. Since the DSM operates at the level of the disk driver, it can additionally load balance traffic across routes based on the number of data blocks being processed. This sixth policy, named *Least Blocks*, will route each subsequent request down the path that has the fewest data blocks being processed.

It is important to note that policies with MPIO are applied to individual devices (LUNs), enabling each connected LUN to be assigned its own policy based on need. This behavior is different than with MCS, where each LUN that is exposed into a single session must share the same policy.

Note

Be aware that MPIO and MCS both achieve link redundancy using a protocol that exists above TCP/IP in the network protocol stack, while NIC teaming uses protocols that exist below TCP/IP. For this reason, each individual MPIO or MCS connection requires its own IP address that is managed within the iSCSI Initiator Control Panel. This is different than with NIC teaming, where ports are aggregated via the switching infrastructure and a single IP address is exposed.

Which Option Should You Choose?

It is commonly considered that MPIO and MCS are relatively similar in their level of performance and overall manageability. Microsoft's MPIO tends to use fewer processor resources than MCS, particularly under heavy loads; however, MCS tends to have slightly better performance as long as the number of connections per session remains low.

With this in mind, consider the following guidelines when determining which option for storage connection redundancy you should choose for your Hyper-V environment:

- Traditional NIC teaming is not considered a best practice for storage connections.
- Some storage devices do not support the use of MCS. In these cases, your only option is to use MPIO.
- Use MPIO if you need to support different load-balancing policies on a per-LUN basis. This is suggested because MCS can only define policies on a per-session basis, while MPIO can define policies on a per-LUN basis.
- Hardware iSCSI HBAs tend to support MPIO over MCS as well as include other features such as Boot-from-iSCSI. When using hardware HBAs, consider using MPIO.
- MPIO is not available on Windows XP, Windows Vista, or Windows 7. If you need to create iSCSI direct connections to virtual machines, you must use MCS.
- Although MCS provides a marginally better performance over MPIO, its added processor utilization can have a negative impact in high-utilization Hyper-V environments. For this reason, MPIO may be a better selection for these types of environments.

Do I Need Hardware iSCSI HBAs?

This guide has talked extensively about the use of traditional server NICs as the medium for iSCSI network traffic. However, specialized hardware HBAs for iSCSI traffic exist as add-ons. These specialized devices are dedicated for use by iSCSI connections and potentially provide a measure of added performance over traditional network cards.

As such, you may be asking "Do I need these special cards in my Hyper-V servers?" Today's conventional wisdom answers this question with, "perhaps not."

iSCSI network processing represents a relatively small portion of the overall processing of SCSI disk commands in Windows, with the majority of processing occurring in the network stack, kernel, and file system. Windows Server 2008 in cooperation with server-class NIC vendors now includes support for a number of network optimizations (TCP Chimney, Receive Side Scaling, TCP Checksum Offload, Jumbo Frames) that improve the overall processing of network traffic, and therefore iSCSI processing as well.

One traditional configuration where hardware iSCSI HBAs have been necessary was when Boot-from-iSCSI was desired. These HBAs have typically included the necessary pre-boot code needed to boot a server from an iSCSI SAN. However, today's production NICs found in your everyday servers are beginning to natively support Boot-from-iSCSI, further driving the answer to this question towards a resounding "no."

Getting to High Availability with Hyper-V

All of this discussion prepares us to answer the primary question: *How does one achieve high availability with Hyper-V and iSCSI?* With all the architectural options available, answering this question best requires a bit of an iterative approach. That iterative approach recognizes that every implementation of Hyper-V that desires true high availability must do so via the Windows Failover Clustering feature.

This general-purpose clustering solution enables Windows Servers to add high availability to many different services, Hyper-V being only one in its long list. Thus, being successful with highly-available Hyper-V also requires skills in Windows Failover Clustering. While the details of installing and working with Windows Failover Clustering are best left for other publications, this chapter and guide will assist with correctly creating the needed storage and networking configurations.

The second point to remember is that Windows Failover Clustering requires the use of shared storage between all hosts. Using Cluster Shared Volumes (CSV) in Windows Server 2008 R2, this storage is actively shared by all cluster nodes, with all nodes accessing connected LUNs at once. Microsoft's CSV transparently handles the necessary arbitration between cluster nodes to ensure that only one node at a time can interact with a Hyper-V virtual machine or its configuration files.

Going a step further, Hyper-V and its high-availability clusters can obviously be created in many ways, with a range of redundancy options available depending on your needs, available hardware, and level of workload criticality. Obviously, the more redundancy you add to the environment, the more failures you can protect yourself against, but also the more you'll spend to get there.

It is easiest to visualize these redundancy options by iteratively stepping through them, starting with the simplest options first. The next few sections will start with a very simple single-server implementation that enjoys redundant connections. Through the next set of sections, you'll see where additional levels of redundancy can be added to protect against various types of failures.

To keep the figures simple, color-coding has been used for the connections between server and network infrastructure. That color-coding is explained in Figure 2.7. As you can see, Production Network (or, "virtual machine") connections are marked in green, with Storage Network connections marked in red. It is a best practice as Hyper-V clusters are scaled out to separate management traffic from virtual machine traffic, and where appropriate, it is labeled in black. It is also recommended that the cluster itself be reserved a separate network for its heartbeat communication. That connection has been labeled in blue where appropriate.

Figure 2.7: Color coding for the following set of figures.

Single Server, Redundant Connections

The simplest of configurations involves creating a single-server Hyper-V environment (see Figure 2.8). Here, a single server connects to its network via a single network switch. This configuration is different from the overly-simplistic diagram first seen in Figure 1.1 in that both the Production Network and Storage Network connections have been made redundant in the setup in Figure 2.8.

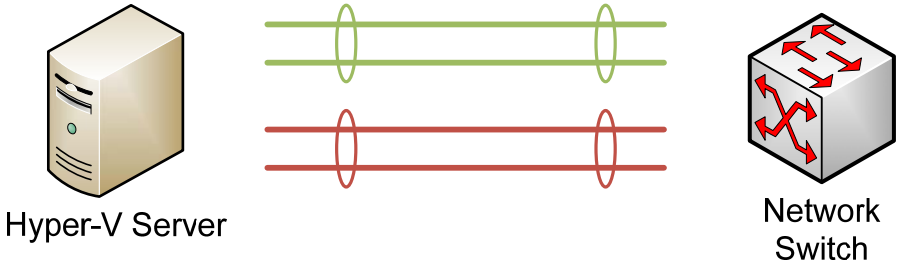

Figure 2.8: Single Hyper-V server, redundant connections.

In this architecture, Hyper-V server traffic is segregated into two different subnets. This is done to separate storage traffic from production networking traffic, and is an important configuration because of Hyper-V's very high reliance on its connection to virtual machine storage. Separating traffic in this manner ensures that an overconsumption of traditional network bandwidth does not impact the performance of running virtual machines.

Both connections in this architecture have also been made highly redundant, though through different means. Here, Production Network traffic is teamed using a network switching protocol such as 802.3ad NIC teaming, while Storage Network traffic is aggregated using MPIO or MCS.

Single Server, Redundant Path

Although this first configuration eliminates some points of failure through its addition of extra connections, the switch to which those connections occur becomes a very important single point of failure. Should the switch fail, every Hyper-V virtual machine on the server will cease to operate.

To protect against this situation, further expansion of connections can be made to create a fully-redundant path between the Hyper-V server and the production network core as well as between Storage Network NICs and the storage device. Figure 2.9 shows how this might look.

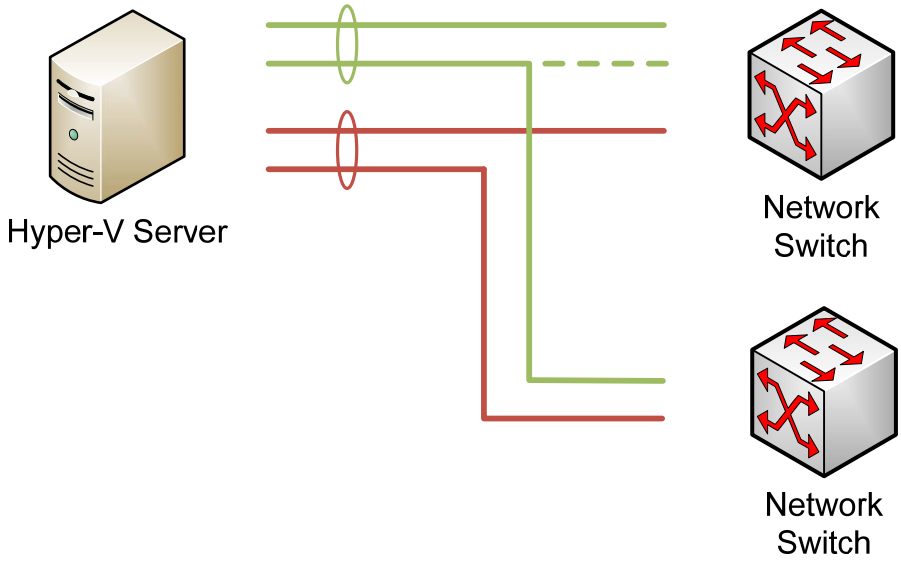

Figure 2.9: Single Hyper-V server, fully-redundant path.

In this configuration, Production Network connections for the single Hyper-V server can either remain in their existing configuration to the single network switch or they can be broken apart and directed to different switches. This option is presented here and elsewhere with an "optional" dashed line because not all networking equipment can support the aggregating of links across different switch devices.

This limitation with NIC teaming highlights one of the benefits of MPIO and MCS. Due to these protocols' position above TCP/IP, each Storage Network connection leverages its own IP address. This address can be routed through different paths as necessary with the protocol reassembling data on either end.

> **Note**
> It is also important to recognize in any redundant path configuration that a true "redundant path" requires separation of traffic at every hop between server and storage. This requirement can make redundant pathing an expensive option when supporting networking equipment is not already in place.

Hyper-V Cluster, Minimal Configuration

Yet even the most highly-available network path doesn't help when a Hyper-V server's motherboard dies in the middle of the night. To protect against the loss of an individual server, Hyper-V must run atop a Windows Failover Cluster. This service enables virtual machines to be owned by more than one server, as well as enables the failover of virtual machine ownership from one host to another.

As such, creating a Hyper-V cluster protects against an entirely new set of potential failures. Such a cluster, see Figure 2.10, requires that all virtual machines are stored elsewhere on network-attached disks, with all cluster nodes having concurrent access to their LUN.

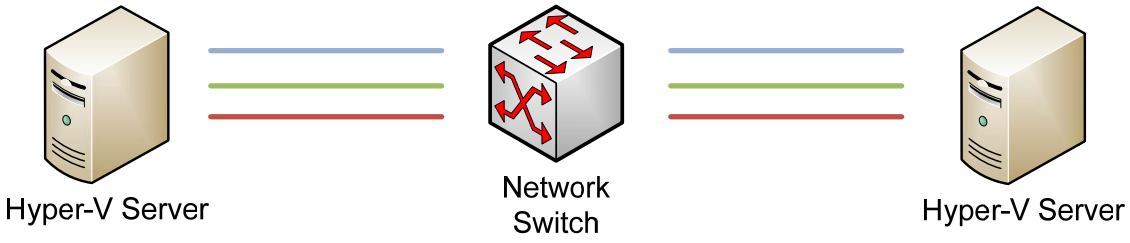

Figure 2.10: Hyper-V Cluster, minimal configuration.

Microsoft's recommended minimum configuration for an iSCSI-based Hyper-V cluster (or, indeed any iSCSI-based Windows Failover Cluster) requires at least three connections that exist on three different subnets. Like before, one connection each is required for Production Network and Storage Network traffic. A third connection is required to handle inter-cluster communication, commonly called the "heartbeat." This connection must be segregated due to the low tolerance for latency in cluster communication.

Hyper-V Cluster, Redundant Connections

Such a cluster configuration like the one previously explained actually removes points of redundancy as it adds others. Although the previous configuration has the potential to survive the loss of a cluster host, its configuration is no longer redundant from the perspective of the network connections coming out of each server.

Needed at this point is the merging of the redundant configuration from the single-server configuration with the now clustered configuration. That updated architecture is shown in Figure 2.11. There, both Production Network and Storage Network connections have been made redundant to the network switch using the protocols explained earlier.

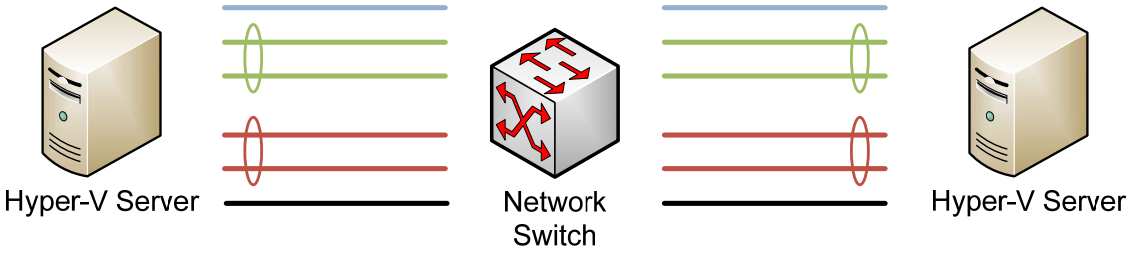

Figure 2.11: Hyper-V cluster, redundant connections.

You'll also see in Figure 2.11 that an additional black line has been drawn between both Hyper-V servers and the network switch. This line represents an additionally-segregated network connection that is used for managing Hyper-V. It is considered a best practice with mature Hyper-V implementations to segregate the management of a Hyper-V server from the networks used by its virtual machines. This is done for several reasons:

- Segregation of security domains—Virtual machines operate at a security trust level that is considered higher than that of management traffic. By segregating virtual machine traffic from management traffic, virtual machines can be better monitored. Further, management connections cannot be used to intercept virtual machine communications.
- Segregation of Live Migration traffic—Transferring ownership of a virtual machine from one host to another can consume a large amount of available bandwidth over a short period of time. This consumption can have a negative impact on the operations of other virtual machines. By segregating Live Migration traffic into its own subnet that is shared with management traffic, this effect is eliminated.
- Protection of management functionality—In the case where a network attack is occurring on one or more virtual machines, segregating management traffic ensures that the Hyper-V host can be managed while troubleshooting and repair functions are completed. Without this separate connection, it can be possible for a would-be attacker to deny access to administrators to resolve the problem.

Hyper-V Cluster, Redundant Path

Finally, this discussion culminates with the summation of all the earlier architectures, combining redundant paths with a fully-functioning Hyper-V cluster. This architecture, see Figure 2.12, enjoys all the benefits of the previous iterations all at once, yet requires the greatest number of connections as well as the largest amount of complexity.

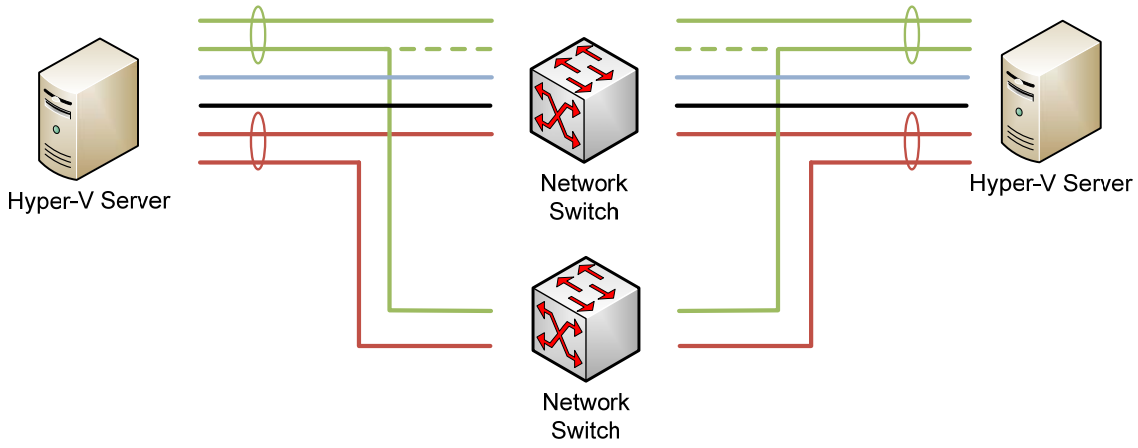

Figure 2.12: Hyper-V cluster, fully-redundant path.

It should be obvious to see the level of added cost that an environment such as this brings. Each cluster node requires a minimum of six network connections, spread across two halves of a switching infrastructure. Due to their inability to share RAM resources between running virtual machines, Hyper-V clusters operate with the greatest levels of efficiency when they are configured with more nodes than less. Thus, a four-node cluster will require 24 connections, with an eight-node cluster requiring 48 connections, and so on.

High Availability Scales with Your Pocketbook

With all this added expenditure comes the protection against many common problems. Individual servers can fail and virtual machines will automatically relocate elsewhere. Disks can fail and be automatically replaced by storage device RAID. Individual connections can fail with the assurance that surviving connections will maintain operations. Even an entire switch can fail and keep the cluster active. It is important to recognize that your level of need for high availability depends on your tolerance for loss. As with physical servers, more redundancy options costs you more but ensures higher reliability.

But reliability in Hyper-V's storage subsystem is fundamentally critical as well. If you create all these connections but attach them to a less-than-exemplary SAN, then you've still set yourself up for failure. Finding the right features and capabilities in such a storage device is critical to maintaining those virtual machine disk files as they're run by cluster nodes.

The next chapter of this book takes a step back from the Hyper-V part of a Hyper-V architecture, and delves deep into just those capabilities that you probably will want in your SAN infrastructure. It will discuss how certain SAN capabilities being made available only today are specifically designed to provide an assist to virtualization infrastructures.

Download Additional eBooks from Realtime Nexus!

Realtime Nexus—The Digital Library provides world-class expert resources that IT professionals depend on to learn about the newest technologies. If you found this eBook to be informative, we encourage you to download more of our industry-leading technology eBooks and video guides at Realtime Nexus. Please visit http://nexus.realtimepublishers.com.